부분 중복 데이터베이스에서 중복 데이터의 트리를 이용한 일관성 유지

배 미 숙† · 황 부 현††

요 약

데이터의 중복은 가용성과 시스템의 성능을 향상시키기 위하여 사용한다. 분산 데이터베이스 시스템에서는 데이터베이스의 일관성뿐만 아니라 중복 일관성도 유지해야 한다. 이 논문에서는 부분 중복 환경에서 각 데이터 마다 주사본을 루트로 하여 그 데이터의 사본들을 계층적인 트리로 조직화하고, 이 트리를 이용하여 연산들의 충돌문제를 해결하는 효율적인 알고리즘을 제안한다. 각 데이터의 계층구조화는 갱신이 데이터의 계층구조를 따라 사본이 갖는 사이트에만 전파될 수 있으므로 갱신의 불필요한 전파시간을 제거하여 갱신 전달 시간을 줄일 수 있다. 그리고 제안한 알고리즘은 지연 갱신 전파로 인하여 갱신 전파 도중에 발생할 수 있는 충돌로 인한 비직렬성 문제를 타임스탬프와 보상 트랜잭션을 이용하여 해결한다. 이러한 방식의 연산들의 충돌문제의 해결은 갱신전파 시간을 줄이면서도 일관성 있는 데이터를 제공할 수 있다.

The Consistency Management Using Trees of Replicated Data Items in Partially Replicated Database

MiSook Bae† · Buhyun Hwang††

ABSTRACT

The replication of data is used to increase its availability and to improve the performance of a system. The distributed database system has to maintain both the database consistency and the replica consistency. This paper proposes an algorithm which resolves the conflict of the operations by using the mechanism based on the structure that the replicas of each data item are hierarchically organized. Each update is propagated along the tree based on the fact that the root of each data item is the primary replica in partially replicated databases. The use of a hierarchy of data may eliminate useless propagation since the propagation can be done only to sites having the replicas. In consequence, the propagation delay of updates may be reduced. By using the timestamp and a compensating transaction, our algorithm resolves the non-serializability problem caused by the conflict of operations that can happen on the way of the update propagation due to the lazy propagation. This resolution also guarantees the data consistency.

키워드 : 부분 중복 데이터베이스(Partially Replicated Databases), 데이터의 계층구조화(Hierarchy of Data), 타임스탬프(Timestamp), 중복 일관성(Replication Consistency)

1. 서 론

분산 데이터베이스 시스템에서는 서버 부하의 균형화, 시스템의 가용성과 신뢰성을 위하여 데이터 중복을 사용한다. 이러한 시스템의 예로는 그룹웨어, 라우팅 테이블, 데이터웨어하우스를 들 수 있다. 중복 일관성을 유지하기 위해서는 갱신 전파가 필요하다. 중복 데이터베이스 시스템은 데이터의 중복 정도에 따라 완전 중복 환경과 부분 중복 환경으로 분류할 수 있다. 지금까지 중복 환경을 다룬 대부분의 연구들은 완전 중복 환경에서 제안되었다[1-5]. 그러나 완전 중복 환경에서 제안된 방법을 부분 중복 환경에 그대로 적용할 경우 불필요한 메시지의 발생으로 인한 시스템 성능이 저하되는 문제점이 있다.

또한, 중복 환경에서는 다른 사이트의 사본들을 갱신하는 방법에 따라 일관성 유지가 문제가 된다. 중복된 데이터를 접근하는 트랜잭션의 정확성을 위해 대부분의 시스템에서는 1-사본 직렬성(1SR : one-copy serializability)을 사용한다. 그리고 데이터의 중복 일관성을 위해 갱신을 전파하는 방법으로는 지연갱신전파(lazy propagation) 방법과 즉시갱신전파(eager propagation)방법이 있다[6, 7]. 즉시갱신전파 방법은 갱신 전파를 위해 모든 노드의 사본들을 하나의 트

※ 본 논문은 한국과학재단 지역대학우수과학자지원연구(R05-2003-000-10532-0)에 의하여 지원되었음.
† 정 회 원 : 전남대학교 대학원 전산학과
†† 정 회 원 : 전남대학교 전산학과 교수
논문접수 : 2002년 11월 13일, 심사완료 : 2003년 4월 7일

랜잭션으로 동기화시킨다. 하지만 사본들의 동기화로 인한 갱신 성능의 저하와 트랜잭션의 응답 시간이 길어지는 단점이 있다. 지연갱신전파 방법은 갱신 트랜잭션의 완료 후에 갱신을 비동기적으로 전파하는 방법이다. 이로 인하여 모든 노드에 갱신이 반영되기를 기다리지 않아도 되므로 트랜잭션의 응답 시간은 빨라지나 갱신 도중에 탐지되는 갱신 충돌이 발생할 수 있다. 즉 완료된 갱신 전파 연산과 충돌되는 읽기 연산의 비직렬성이 문제가 된다.

이 논문에서는 부분 중복 환경에서의 중복 관리 방법을 제안한다. 완전 중복 환경에서 관리 방법을 부분 중복 환경에 적용하면 모든 사이트에 갱신 결과를 전파시킴으로써 불필요한 메시지 전송이 발생하고, 전파 시간 또한 길어진다. 이를 방지하기 위하여 부분적으로 중복된 데이터와 사이트를 갖는 구조에서 중복된 데이터가 있는 사이트를 계층화하여 갱신이 발생할 때 불필요한 메시지가 발생하지 않고 빠른 시간내에 갱신을 전파할 수 있는 메카니즘을 제안한다. 그리고 이 논문에서는 갱신 전파를 위해 지연갱신 전파 방법을 사용한다. 이 방법은 트랜잭션이 갱신을 수행한 후 전파되는 시점과 임의의 사이트에서 트랜잭션 연산의 완료 시점의 차이로 인하여 비직렬성 문제가 발생할 수 있는데 이러한 비직렬성 문제를 타임스탬프를 이용하여 해결한다. 또한, 이 논문은 갱신이 드문 환경을 가정하며 단일 마스터 갱신 기법을 사용한다.

이 논문의 구성은 다음과 같다. 2장에서는 중복 환경에서 중복 관리를 제안한 연구들에 대하여 살펴보고, 3장에서는 기존의 연구에서 발생하는 문제점을 지적한다. 4장에서는 부분 중복 환경에서 중복 관리를 위한 시스템 모델과 알고리즘을 제안하며, 5장에서는 결론을 기술한다.

2. 관련 연구

이 절에서는 중복 환경에서 중복 관리 방법을 제안한 기존의 연구에 대하여 살펴본다.

[6-10]에서는 지연 마스터 전파 중복 데이터베이스(lazy master replication database)에서 지연 전파로 인하여 중복 일관성(replication consistency)이 위배될 수도 있다는 것을 지적하고 있다. 특히, [8, 9]는 중복 일관성을 유지하기 위해 빠른 전파 방법인 즉시전파(immediate-propagation) 방법을 제안하였다. 또한, 상호 일관성(mutual consistency)의 완화와 최신 데이터에 대한 일탈 정도를 나타내는 신선도(freshness)의 개념을 이용하여, 2PC(2 phase commit protocol)의 성능 저하를 지적하면서 신선도를 개선하는 방법을 제시하였다. 그러나 이것은 신뢰적인 멀티캐스팅 프로토콜이 지원되는 환경이어야 한다는 한계를 갖는다. [4]는 연산을 전파하는 것이 아니라 이전 내용과 현재 내용과의 차이인 델타를 전달함으로써 연산 충돌 문제를 다루지 않고 있다. [6]은 이동 트랜잭션에 적용 가능한 2단계 중복 알고리즘을 제안하였다. [10]은 전역 트랜잭션과 서브 트랜잭션을 구분하여 전역 트랜잭션사이의 충돌을 줄이기 위해 가상 사이트(virtual site)개념과 중복 그래프를 이용한 프로토콜을 제시하였다.

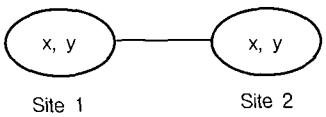
중복 데이터베이스는 일관성을 유지하기 위해서 1-사본 직렬성을 보장해야 한다. 1-사본 직렬성은 중복 데이터에서의 트랜잭션의 수행 결과가 단독의 데이터에서 한번 수행된 결과와 같아야 함을 의미한다. 데이터가 여러 사이트에 중복 저장된 경우에는 데이터의 일관성 유지를 위한 갱신 전파 방법이 필요하다. [6, 7, 11, 12]에서는 이러한 갱신을 전파하는 방법으로 즉시갱신 전파 기법과 지연갱신 전파 기법을 언급하고, 갱신을 제어하는 방법을 그룹(group) 갱신 방식과 마스터(master) 갱신 방식으로 분류하였다. 갱신이 어느 곳에서나 발생할 수 있는 그룹 갱신 방식은 주사본이 있는 곳에서만 갱신이 가능한 마스터 갱신 방식에 비해 갱신 충돌의 가능성이 더 많다.

[12]에서는 대규모 중복 데이터베이스 시스템에서 강한 일관성을 유지하는 데에 대한 문제점을 언급하였다. 많은 수의 사이트를 갖는 시스템이 모두 동일한 상태를 갖게 하기 위한 부담을 줄이기 위해 계층구조를 갖는 노드 구조를 기반으로 한 비동기 갱신 전파 방법을 제안했다. 각 노드들을 지형적 위치나 논리적 기준에 의하여 클러스터로 그룹화하고, 클러스터들을 계층 구조화하여 일관성 유지 기준을 완화하였다. 결과적으로 트랜잭션의 철회율이 줄어들고, 사용자 수준의 트랜잭션 응답 시간을 개선할 수 있다. 또한 일관성의 정도에 따라 다양한 뷰를 허용할 수 있다. 부가하여 이 연구에서는 일부 결과가 직렬 일관성(sequential consistency)을 확보하지 못하더라도 합리적인 응답시간을 지닌 서비스에 대한 접근을 허용하는 고가용성 서비스(highly available services)를 위한 약한 일관성의 필요성을 언급하였다[11, 12]. 또한, 많은 사본들이 있는 대규모 시스템에서는 관리자의 융통성을 증가시키기 위해 사본들의 계층적인 조직화가 필요함을 지적했다. 근접한 사이트들을 그룹으로 분할함으로써 관리자는 전역적으로 시스템의 재구성을 하지 않고 사본의 추가, 삭제, 다른 위치로의 이동이 가능하기 때문이다[13].

3. 문제 정의

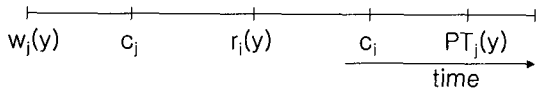
이 절에서는 중복 데이터베이스에서 발생할 수 있는 지연 갱신 전파로 인한 직렬가능하지 않은 수행에 대한 예를 기술하고 계층구조화의 필요성을 언급한다.

(예 1) 지연 갱신 전파로 인한 비직렬성의 예



두 개의 데이터 항목 x, y가 위의 그림처럼 site 1과 site 2에 중복되어 있다고 가정하자. 여기에 두 개의 트랜잭션 $T_1 : w_1(x)\ r_1(y)$과 $T_2 : r_2(x)\ w_2(y)$가 site 1과 site 2에서 각각 수행되고, x의 주사본은 site 1, y의 주사본은 site 2에 존재한다고 가정하자. 갱신 연산은 반드시 주사본 사이트에서 수행되어야 하며, 판독 연산은 임의의 사이트에서 수행 가능하다. 이 때 T_1이 site 1에서, T_2가 site 2에서 수행되어 완료될 수 있다. 갱신을 전파하기 위해 site 1은 site 2로 x에 대한 갱신을 전파하고, site 2는 y에 대한 갱신을 site 1으로 전파한다. 결국 site 1에서 가능한 하나의 히스토리 H1은 $w_1(x)\ r_1(y)\ c_1\ w_2(y)$ 이고($T_1 \rightarrow T_2$), site 2의 히스토리 H2는 $r_2(x)\ w_2(y)\ c_2\ w_1(x)$가 되어($T_2 \rightarrow T_1$) 전체 직렬화 순서에 사이클이 발생하게 된다. □

(예 2) 지연 갱신 전파로 인한 비직렬성의 예



두 번째 예는 지연 갱신 전파 방법에서 트랜잭션이 갱신을 수행한 후 완료한 시점과 전파되는 시점의 차이로 인하여 발생하는 문제에 관한 예이다. y를 갱신한 임의의 트랜잭션 T_j가 완료되면 y에 대한 갱신을 y가 저장된 모든 사이트에 전파해야 한다. y가 저장되어 있는 사이트에 y에 대한 갱신이 도착하기전, T_j 보다 타임스탬프가 큰 트랜잭션 T_i가 y를 판독한 후 완료되었다면 T_i가 판독한 값은 올바른 값이라 할 수 없다. 여기서 PT_j는 T_j에 대한 전파 트랜잭션의 의미이다. 타임스탬프 규칙에 의하면 PT_j는 철회되어야 하지만 이미 완료된 트랜잭션이므로 PT_j의 철회는 불가능하다. □

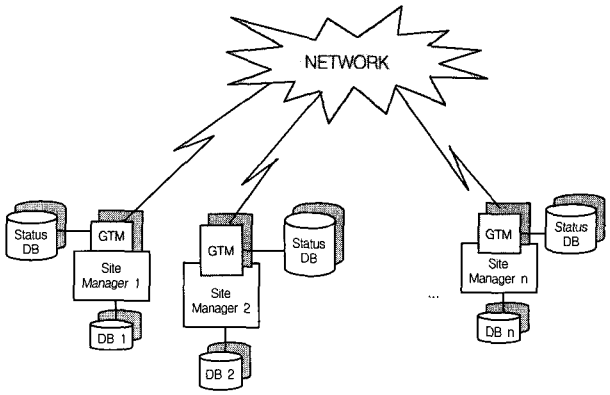
중복 데이터베이스 시스템의 유형 중 대규모 시스템에서는 탐색 시간과 확장성 면에서 선형구조보다 계층구조가 더 우수하다. 이러한 계층구조에서 마스터 갱신 기법을 적용하는 경우를 고려해보자. 그 계층구조의 최상위 노드는 주사본이 있는 노드이고, 갱신은 반드시 주사본에서 수행된다. 이 때 갱신 트랜잭션이 최상위 레벨이 아닌 임의의 레벨에 있는 사이트에 제출되면 주사본 사이트로 옮겨가서 갱신이 수행된 뒤 하위 레벨로 전파시켜야 한다. 이러한 구조에서 level-depth가 클 경우 갱신 연산을 수행할 때 주사본 사이트를 찾는데 걸리는 시간과 갱신 전파 시간이 길어진다. 만약 갱신 트랜잭션이 최하위 레벨에 있는 사이트로 제출된다면 최상위 레벨의 주사본 사이트(primary site)를 찾는데 드는 오버헤드가 발생하며, 이러한 구조는 완전 중복 환경을 고려한 것이므로 중복 데이터를 갖고 있지 않은 사이트까지도 갱신 메시지를 받게 되어 비효율적이다. 그러므로 데이터의 계층구조를 따라 사본을 갖고 있는 사이트에만 전파될 수 있는 메카니즘이 필요하다.

4. 부분 중복 환경에서 중복 관리를 위한 시스템 모델과 알고리즘

이 절에서는 제안하는 방법인 UPP-TRD(Update Propagation Protocol using Tree of Replicated Data items)에 대한 알고리즘과 실행 예를 기술한다.

4.1 시스템 모델

이 논문에서 사용하는 시스템 모델은 (그림 1)과 같다. 모델은 크게, 각 사이트에 존재하는 전역 트랜잭션 관리자(GTM)와 사이트 관리자(site manager)로 구성된다. 각 데이터베이스의 데이터는 부분 중복되어 있다. 전역 트랜잭션 관리자는 트랜잭션에 대한 타임스탬프를 부여하고, 조정자가 되며 여러 사이트를 접근하여 수행하는 트랜잭션의 분배를 담당한다. 또한, 상태 데이터베이스(Status DB)에는 갱신 연산이 수행될 주사본 사이트의 위치를 유지한다. 각 데이터의 주사본이 위치한 사이트의 관리자는 다른 사본들의 위치를 트리 형태로 유지하여 효율적으로 갱신을 전파한다. 중복 트리를 이용하여 각 사이트들은 [12, 13]에서 사용한 방법처럼 갱신을 효율적으로 전파할 수 있도록 사이트 구조가 재구성될 수 있다.



(그림 1) 시스템 모델

알고리즘 설명을 위해 제출된 사이트에서 실행되는 트랜잭션을 주 트랜잭션이라 하며, 주 트랜잭션이 완료된 후 갱신 전파를 위해 사본이 있는 다른 사이트로 보내어져 실행되어지는 트랜잭션을 전파 트랜잭션이라 한다.

4.2 가정 및 자료구조

이 논문에서는 다음과 같은 사항을 가정하고 있다.

첫째 : 데이터는 부분 중복되어 있다.

둘째 : 네트워크는 신뢰성이 있다. 즉, 한 사이트에서 다른 사이트로 전달한 메시지는 반드시 전달되며, 각

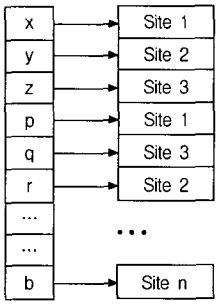
메시지들은 보내진 순서대로 도착한다.

셋째 : 중복된 각 사본에 대하여 다중 뷰를 허용한다. 이는 갱신의 전파가 항상 모든 사본에 동시에 적용되지 않기 때문에 어느 정도의 데이터 비일관성을 허용한다는 의미이다.

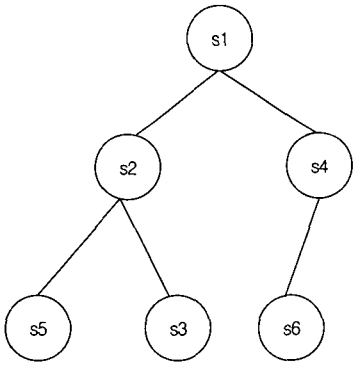
넷째 : 중복된 데이터의 갱신을 위해 마스터 갱신 방법(master-update)을 사용한다. 데이터의 갱신은 주사본(primary copy)에서만 수행되며, 트랜잭션이 완료되면 갱신이 전파된다.

다섯째 : 각 트랜잭션에 유일한 타임스탬프를 부여하기 위해 각 사이트의 클럭이 동기화되어 있다.

전역 트랜잭션 관리자가 유지하는 상태 데이터베이스와 주사본 사이트에서 유지하는 중복 데이터 트리 구조에 대한 예는 (그림 2), (그림 3)과 같다. 상태 데이터베이스에는 (그림 2)와 같이 각 데이터 항목에 대한 주사본의 위치를 유지한다. 또한 각 사본의 주사본 사이트에는 그의 사본이 있는 사이트에 대한 정보를 트리 형태로 유지하여 갱신의 전파를 효율적으로 할 수 있다. 임의의 주사본 사이트에 유지되는 트리 구조의 예는 (그림 3)과 같다. 트리 구성에 대한 자세한 내용은 4.3절에서 다룬다.



(그림 2) 상태 데이터베이스의 예



(그림 3) 임의의 데이터에 대한 트리 구조의 예

4.3 UPP-TRD 알고리즘

이 절에서는 부분 중복 데이터베이스에서의 트랜잭션 수행 방법과 트리 구조를 이용한 갱신 전파 방법에 대하여 기술한다.

4.3.1 상태 데이터베이스의 구성 및 사이트 재구성

전역 트랜잭션 관리자(GTM)는 상태 데이터베이스를 구성하기 위해 각 데이터 항목에 대한 주사본의 정보를 수집한다. 각 데이터 항목에 대한 주사본 사이트는 데이터에 대한 접근 횟수가 가장 많은 사이트이다. 사본들은 계층적인 트리 구조 형태를 갖는다. 트리의 루트는 데이터의 접근 횟수가 가장 높은 사이트로 하며, 접근횟수가 비슷하면서 근접한 사이트들은 트리에서 같은 레벨에 위치하도록 구성한다. 이러한 형태는 같은 부모를 갖는 같은 레벨의 사이트들을 그룹으로 분할하여 작업을 가능하게 한다. 또한 효율적인 갱신 전파를 위해 중복 데이터에 대한 트리 구조와 동일하게 사이트 구조를 재구성한다.

4.3.2 주트랜잭션의 제출 및 타임스탬프 부여

사용자 트랜잭션은 임의의 사이트의 전역 트랜잭션 관리자에 제출되며, 트랜잭션이 제출된 사이트가 조정자가 된다. 각 트랜잭션의 조정자는 각 트랜잭션에 유일한 타임스탬프를 부여한다. 타임스탬프는 지역 타임스탬프와 사이트 번호로 구성된다. 지역 타임스탬프는 트랜잭션이 제출된 사이트의 시간 값이며, 사이트 번호는 지역이 다른 동일한 타임스탬프를 식별하기 위해 사용되어진다. 또한, 사이트 관리자는 제출된 트랜잭션들을 갱신 트랜잭션과 판독 전용 트랜잭션으로 구분한다.

4.3.3 연산의 스케줄

트랜잭션의 갱신 연산은 반드시 주사본 사이트로 전달되어 수행되며, 판독 연산은 주사본 사이트에서도 수행을 허용하나, 연산들 사이의 충돌로 인한 지연이 많거나 오버헤드가 클 경우에는 다른 임의의 사이트에서 수행하도록 한다. 또한, 판독 연산은 제출된 사이트에서 수행 가능하나 제출된 사이트에 데이터 사본이 없는 경우에는 사본이 존재하는 사이트 중 접근 비용이 가장 적거나 부하가 적어서 연산을 빨리 수행할 수 있는 사이트 중 하나로 결정할 수 있다.

4.3.4 주트랜잭션의 완료

분산된 사이트에서 수행된 트랜잭션의 각 트랜잭션의 조정자는 트랜잭션의 완료를 위해 2PC 프로토콜을 사용한다. 판독 연산만으로 구성된 트랜잭션은 완료시 수행중인 사이트가 주사본 사이트가 아닌 모든 데이터의 주사본 사이트에 완료했음을 알린다. 이것은 판독 연산이 수행된 후에 수행되는 갱신 연산이 타임스탬프 순서를 유지하는지를 검사하기 위해서이다. 판독 연산과 갱신 연산은 충돌 관계이기 때문에 그들 사이의 타임스탬프 순서를 유지해야 직렬가능

한 히스토리를 만들 수 있다. 갱신 연산을 포함하고 있는 트랜잭션이라면 주사본 사이트에서 전파 트랜잭션이 수행되도록 트리거를 제공한다.

4.3.5 갱신의 전파

갱신 트랜잭션이 완료되면 조정자는 그것의 갱신 연산을 주사본 사이트에 제출한다. 이것은 트랜잭션의 제출과 완료에 대한 책임이 트랜잭션이 제출된 조정자 사이트에 있기 때문이다. 조정자는 트랜잭션의 모든 연산을 수행한 후 2PC를 통해 완료 여부를 결정하고, 트랜잭션이 데이터를 갱신했으면 갱신 전파가 수행될 수 있도록 해야하는데 조정자 사이트와 주사본 사이트가 일치하지 않을 경우에는 주사본 사이트에 결과를 알려주어야 한다.

갱신연산을 수행한 후, 주사본 사이트는 트리 구조에서 그 주사본 사이트의 자식에게로 갱신 결과를 전파한다. 완료된 트랜잭션의 타임스탬프가 전파되어질 갱신 연산에 할당된다. 임의의 사이트에 제출되어진 주트랜잭션 T_i와 전파되어오는 전파 트랜잭션 PTj의 갱신 연산의 타임스탬프를 비교하여 T_i의 타임스탬프가 PT_j의 타임스탬프보다 크며, T_i가 이미 종료됐고 접근 데이터 항목이 일치하면, 그 트랜잭션 T_i의 갱신 연산을 보상(compensation)한 후 재수행되며, 그렇지 않으면 정상 수행된다. 재수행은 새로운 타임스탬프를 할당받으며 새로운 트랜잭션으로 처리된다. 판독전용 트랜잭션은 완료하면 자신의 사이트가 주사본 사이트가 아닌 모든 데이터 a에 대해서, a의 주사본 사이트에 완료했음을 알린다.

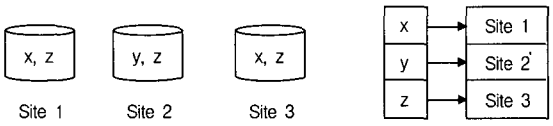
각 데이터 항목의 주사본 사이트에는 사본이 유지된 사이트들의 구조를 트리 형태로 유지한다. 데이터 트리의 루트는 가장 많은 접근 횟수를 가진 사이트이다. 갱신의 결과는 트리를 따라 전파된다. 트리는 전파 시간을 최소화하기 위해 모든 노드까지의 깊이가 동일한 균형 트리(balanced tree)로 유지한다.

제안하는 UPP-TRD 알고리즘은 다음과 같다.

```
1. GTM에 트랜잭션 T가 도착되면,
2. if (T가 주 트랜잭션이면) T에 유일한 타임스탬프 할당 ;
3. for (each operation op(x) of T)
   if (op == r)
      if (현 사이트에서 수행한다면) {
            if (w_ts(x) < ts(T)) perform r(x) ;
            else abort T ;
         }
      if (현 사이트에 부하가 많으면) 적절한 사이트로 op(x) 전송 ;
   if (op == w)
         if (현 사이트가 primary이면)
            if (w_ts(x) < ts(T) && r_ts(x) < ts(T))
               perform w(x) ;  // 임시 저장
         else primary site로 전송 ;
4. for (다른 사이트로부터 전송되어온 각 r(x), w(y)에 대하여)
   if (타임스탬프 규칙을 만족하면) 수행후 결과를 반환 ;
   else 철회 결정 전달 ;
5. if (트랜잭션의 마지막 연산의 수행이 끝나면)
      if (다른 사이트에서 수행된 연산이 존재하면) // 2PC 수행
         수행한 사이트로 vote-req를 보낸다.
         if (결과가 모두 yes) commit T ;
         else abort T ;
6. if (완료한 트랜잭션 T가 write 연산을 포함)
      // 조정자 ≠ 주사본 사이트
      갱신한 데이터의 값을 primary 사이트로 전달
7. if (T가 전파 트랜잭션 PT이면) {  // primary에서 전파를 시작
                                       하기 위한 트랜잭션
      if (PT와 접근 데이터 집합이 공집합이 아니고, 타임스탬프가
         큰 완료된 트랜잭션 Tj가 존재) {
      갱신을 전파할 하위 사이트가 있다면 갱신을 전파 ;
         Tj의 모든 write operation에 대한 compensate operation
         수행 ;
         ts(PT)보다 큰 타임스탬프를 Tj에 할당하여 재수행 }
          // 주트랜잭션
         else PT 수행 // 갱신 반영
      }
```

UPP-TRD 알고리즘의 수행 예는 다음과 같다.

(그림 4)와 같이 3개의 사이트를 가진 부분 중복 데이터베이스 시스템과 각 데이터 항목에 대한 주사본의 위치를 가진 상태 데이터베이스가 구축되어 있다고 가정하자. 즉, 데이터 항목 x의 주사본은 s1이고, y의 주사본은 s2이고, z의 주사본은 s3인 경우이다.



(그림 4) 부분 중복 데이터베이스와 상태 데이터베이스

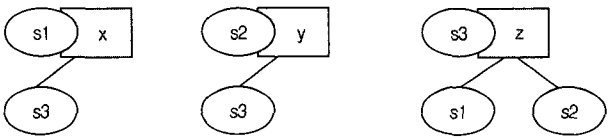
여기에 3개의 트랜잭션이 아래와 같이 각 사이트에서 제출된다고 가정하자.

site 1에서 T_1 : $w_1(x)$ $w_1(y)$

site 2에서 T_2 : $r_2(x)$ $r_2(y)$

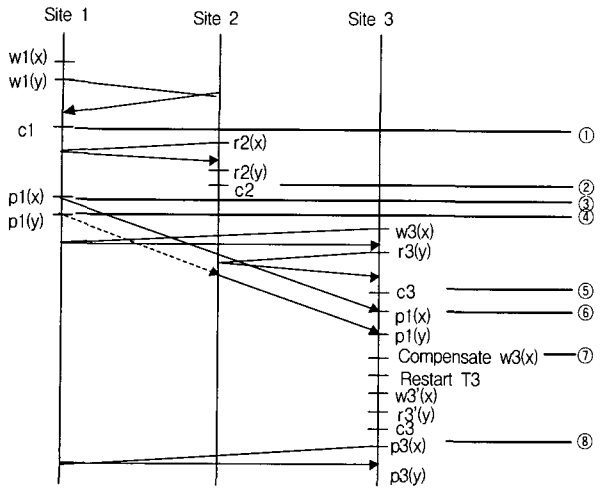
site 3에서 T_3 : $w_3(x)$ $r_3(y)$

각 트랜잭션의 타임스탬프가 $ts(T_1)$ = 10.1, $ts(T_2)$ = 12.2, $ts(T_3)$ = 15.3이라 가정하자. 그리고 주사본이 있는 사이트에서 유지되는 각 중복 데이터의 트리 구조는 (그림 5)와 같다. 트리의 루트는 주사본 데이터 항목을 의미한다.



(그림 5) 주사본에 유지되는 중복 데이터 트리 구조

각 사이트에서 수행되는 트랜잭션의 순서는 (그림 6)과 같다.



(그림 6) 트랜잭션의 수행순서

각 트랜잭션의 갱신 연산은 반드시 주사본이 있는 사이트에서 수행되어야 하며, 판독 연산은 연산을 가장 효율적으로 수행할 수 있는 사이트에서 수행된다. ①번 시점에서는 T_1, ②번 시점에서는 T_2가 각각 완료될 수 있다. ③번 시점에서는 T_1이 완료했으므로 x에 대한 전파가 시작된다. ④번 시점에서도 T_1이 완료했으므로 T_1의 완료를 y에 대한 주사본 사이트에 알림으로써 전파가 시작된다. ⑤번 시점에서는 T_3가 수행되어 완료를 한 시점이다. ⑥번 시점에서는 T_1의 완료에 의해 전파된 연산이 T_3의 완료후에 도착한 경우이다. 그러나 이것은 $ts(T_1) < ts(T_3)$ 이므로 T_3가 T_1보다 먼저 수행되는 것은 타임스탬프 규칙에 어긋난다. 직렬성을 보장하기 위하여 ⑦번 시점에서는 T_3의 쓰기 연산을 보상(compensation)한 후 T_3를 재수행한다. ⑧번 시점에서는 T_3가 ts(p1(x))보다 더 큰 타임스탬프를 할당받아 재수행시켜 전파되는 시점이다. 이 때 각 데이터에 대해 직렬성이 다음과 같이 유지됨을 알 수 있다.

x에 대한 직렬화 순서 : $T_1 \rightarrow T_2 \rightarrow T_3$
y에 대한 직렬화 순서 : $T_1 \rightarrow T_2 \rightarrow T_3$

다음은 제안하는 알고리즘이 직렬가능한 히스토리를 생성함에 대한 증명이다.

[정리] UPP-TRD 방법은 트랜잭션의 직렬성을 보장한다.
(증명) 제안하는 알고리즘에서 직렬성이 위배될 수 있는 경우는 ① 주트랜잭션의 수행이 직렬가능하지 않거나 ② 전파 트랜잭션의 수행이 직렬가능하지 않거나 ③ 주트랜잭션과 전파 트랜잭션의 수행 순서에 비직렬적인 순서가 발생하는 경우이다.

[경우 1] 주 트랜잭션들 사이에서 직렬성이 위배되는 경우

UPP-TRD 방법에 의해 만들어진 각 로컬 스케줄에서 $T_i \rightarrow T_j \rightarrow \cdots \rightarrow T_k \rightarrow T_i$과 같은 순서가 만들어진다고 가정하자. 타임스탬프 규칙에서 임의의 트랜잭션 T_i, T_j에 대하여 $T_i \rightarrow T_j$는 $ts(T_i) < ts(T_j)$임을 의미한다. 그러므로 $T_k \rightarrow T_i$는 $ts(T_k) < ts(T_i)$임을 의미하므로 $T_i \rightarrow T_k$로 가는 순서는 발생하지 않는다.

다음은 $T_i \rightarrow T_j \rightarrow T_k$의 관계에서, T_i가 T_j에 영향을 미칠 수 있는 경우이다.

① T_i가 정상적으로 완료하여 전파까지 성공했다면 문제가 없다.

② T_i가 정상적으로 완료되었으나 전파시 전파 트랜잭션과 충돌되어서, 철회되어 재수행되었다면,
⇒ $T_i \rightarrow T_j$의 에지가 생성되려면, 또 T_j가 영향받으려면 T_i는 갱신 트랜잭션이어야 하며, T_j는 판독전용트랜잭션이나 갱신 트랜잭션이다.

- $T_i \rightarrow T_j$의 에지가 T_i가 갱신 트랜잭션이고, T_j가 판독 트랜잭션일 경우 동일 사이트에서 $ts(T_i) < ts(T_j)$이면 문제가 되지 않고, $ts(T_i) > ts(T_j)$인 경우는 T_j가 철회된다.
  T_i와 T_j가 다른 사이트에서 수행되었다면, T_i의 전파가 도착하기 전까지 T_j는 정상 수행되고, 그들 사이의 타임스탬프 관계에 따라 철회되거나 재수행(보상없이)되므로 연쇄적인 재수행은 없다.
- $T_i \rightarrow T_j$의 에지에서 두 개의 갱신 트랜잭션에 의해 생성되었다고 가정하면, 데이터에 대한 갱신은 반드시 주사본 사이트에서만 수행된다. 하지만 갱신 연산은 반드시 TSO에 의해서만 수행되므로, T_j가 값을 성공적으로 쓰려면 T_i가 완료와 전파를 시작한 다음에야 가능하다. $T_i \rightarrow T_j$일 때 T_i가 전파 트랜잭션에 의해 보상되어진다는 것은 T_i가 완료되었고, 전파를 시작하기 전에 T_j가 이 값에 접근했다는 것을 의미한다. 그런데 이 시점은 주사본 사이트에서 갱신을 한 후 조정자로부터 완료메시지를 받기 전이다. 그러므로 주사본 사이트의 데이터베이스에는 아직 변경이 되지 않은 상태이므로 T_j에 영향을 주지 못한다. 또한, T_i가 active 상태이면 T_j는 철회된다. 따라서 이 경우도 재수행은 발생하지 않는다.

③ Ti가 수행중 철회되었다면 이러한 순서 자체가 만들어지지 않았을 것이므로 문제가 없다.

[경우 2] 갱신 전파 트랜잭션들 사이에서 직렬성이 위배되는 경우

UPP-TRD 방법은 각 메시지들은 보내진 순서대로 도착한다고 가정했기 때문에 임의의 데이터 x에 대한 전파 트랜잭션 $PT_i(x)$, $PT_j(x)$가 있을때 $ts(T_i) < ts(T_j)$ 라면, $PT_i(x) \rightarrow PT_j(x)$가 도착하게 된다. 수행 순서의 역전 현상이

발생할 수 있는 경우는 $ts(T_i) < ts(T_j)$일 때 $PT_j(x)$가 $PT_i(x)$보다 먼저 도착한 경우이다. 이 경우는 [경우 1]에 의해 T_i가 철회되므로 발생하지 않는다. 그러므로 $PT_j(x) \rightarrow PT_i(x)$ 순서는 발생하지 않는다.

[경우 3] 주 트랜잭션과 갱신 전파 트랜잭션 사이에서 직렬성이 위배되는 경우

전파 트랜잭션 $PT_i(x)$와 주트랜잭션 T_j의 연산 $r_j(x)$에서 $ts(T_j) > ts(PT_i)$라면, $r_j(x) \rightarrow PT_i(x)$는 $ts(T_j) > ts(PT_i)$ 하도록 T_j의 타임스탬프가 재할당되고 T_j가 재수행된다. 그러므로 $PT_i(x) \rightarrow r_j(x)$와 동일하게 되므로 직렬성을 유지할 수 있다.

위 세 가지 경우에 대한 증명으로 제안하는 알고리즘이 직렬가능한 히스토리임를 생성함을 보장할 수 있다. □

5. 결 론

중복 데이터베이스는 데이터의 가용성과 성능을 향상시키기 위해 필요하다. 지금까지 제안된 대부분의 중복 관리 방법들은 완전 중복 데이터베이스 시스템을 배경으로 하고 있는데, 비직렬적인 트랜잭션 수행 가능성과 불필요한 전파로 인하여 전파 시간이 길다는 단점이 있다.

최근의 중복 데이터베이스 응용의 한 추세는 시스템의 규모가 커지고 있다는 것이다. 대규모 중복 데이터베이스에서 효율적인 중복 관리를 위해서는 정확한 수행을 보장하기 위한 정확성 검증 방법과 효율적인 갱신 전파 방법이 필요하다. 또한 대규모 시스템에서는 갱신 트랜잭션이 성공적으로 완료된다 하더라도 갱신을 전파하는데 많은 부담이 발생하게 된다.

이 논문에서는 부분 중복 데이터베이스 시스템에 적용될 수 있는 갱신 전파 방법을 제안하였다. 제안하는 방법은 타임스탬프을 사용함으로써 수행되는 트랜잭션의 직렬성을 보장할 수 있고, 전파 트랜잭션과 주 트랜잭션의 연산 충돌이 탐지될 때 보상 트랜잭션을 이용하여 트랜잭션을 재수행시킴으로써 해결할 수 있다. 전파 트랜잭션으로 인한 트랜잭션 재수행의 발생 가능성은 갱신 연산 수에 비례한다. 하지만 이 논문에서는 갱신이 드문 환경을 가정하고 있기 때문에 재수행이 발생할 수는 있으나 그리 많지 않음을 알 수 있다. 중복 데이터에 대한 트리 구조를 이용함으로써 전파 시간을 단축할 수 있다. 트랜잭션 수행에 대한 정확성 기준으로 1-사본 직렬성을 사용하였고, 상태 데이터베이스에 주사본의 위치를 저장함으로써 주사본에 빨리 갱신 연산을 전달할 수 있다. 각 주사본 사이트에는 중복 데이터에 대한 균형된 트리를 유지함으로써 갱신 전달 시간을 단축하였다. 제안 기법은 갱신 비율이 낮은 환경에서 사용자에게 빠른 응답 시간을 제공할 수 있는 비동기 응용에 적합하다.

참 고 문 헌

[1] M. Makpangou, G. Pwwierre, C. Khoury and N. Dorta, "Replicated Directory Service for Weakly Consistent Distributed Caches," IEEE, pp.92-100, 1999.

[2] P. Chundi, D. Rosenkrantz and S. Ravi, "Deferred Updates and Data Placement in Distributed Databases," 12th International Conference on Data Engineering, IEEE, 1996.

[3] R. Jimenez-Peris, M. Patino-Martinez, G. Alonso and S. Arevalo, "Reducing The Latency of Non-Blocking Commitment Using Optimism and Replication," IEEE Int. Conf. on Dependable Systems and Networks(DSN). Goteborg, Sweden, July, 2001.

[4] Y. Saito and H. Levy, "Optimistic Replication for Internet Data Services," 14th International Conference on DISC, Toledo, Spain, pp.297-314, Oct., 2000.

[5] Y. Breitbart, R. Komondoor, R. Rastogi, S. Seshadri and A. Silberschatz, "Update Propagation Protocols for Replicated Databases," In Procs. of ACM SIGMOD International Conf. on Management of Data, Philadelphia, pp97-108, June, 1999.

[6] J. Gary, P. Helland, P. O'Neil and D. Shasha, "The Danger of Replication and a Solution," In Procs. of ACM SIGMOD International Conf. on Management of Data, Montreal, Canada, pp.173-182, 1996.

[7] T. Anderson, Y. Breitbart, H. F. Korth and A. Wool, "Replication, consistency and practicality: Are these mutually exclusive?," In Procs. of the ACM SIGMOD International Conf. on Management of Data, Seattle, WA, pp.484-495, June, 1998.

[8] E. Pacitti and E. Simon, "Update Propagation Strategies to Improve Freshness in Lazy Master Replicated Databases," VLDB Journal, pp.305-318, 2000.

[9] E. Pacitti, P. Minet and E. Simon, "Fast Algorithms for Maintaining Replica Consistency in Lazy Master Replica Databases," Proceedings of The 25th VLDB conference, Edinburgh, Scotland, pp.126-137, 1999.

[10] Y. Breitbart and H. Korth, "Replication and consistency: Being lazy helps sometimes," In Procs. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Tucson, Arizona, pp.173-184, May, 1997.

[11] Coulouris, Dollimore and Kindberg, "Distributed Systems: Concepts and Design," Edition 3, Addison-Wesley, 2001.

[12] X. Liu, B. Heal and E. Du, "Multiview Access Protocols for Large Scale Replication," ACM Transaction on Database Systems, Vol.23, No.2, pp.158-198, June, 1998.

[13] T. Johnson and K. Jeong, "Hierarchical Matrix Timestamps for Scalable Update Propagation," 10th International Workshop on Distributed Algorithm, June, 1996.

배 미 숙

e-mail : msbae@chonnam.chonnam.ac.kr
1989년 전남대학교 전산통계학과(학사)
1993년 전남대학교 전산학과(이학석사)
1999년~현재 전남대학교 전산학과 박사과정
관심분야 : 분산데이터베이스, 분산시스템, 컴퓨터통신

황 부 현

e-mail : bhhwang@chonnam.chonnam.ac.kr
1978년 숭실대학교 전산학과(학사)
1980년 한국과학원(공학석사)
1994년 한국과학원(공학박사)
1981~현재 전남대학교 전산학과 교수
관심분야 : 분산시스템, 객체지향시스템, 이동컴퓨팅